

FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFFFFFFFFFFFFFFF	111	111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111111	111111	XXX	XXX
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFFFFFFFF.FFF	111	111		
FFFFFFFFFFFFFFFF	111	111	XXX	
FFFFFFFFFFFFFFFF	111	111	XXX	
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111		
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	111	111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX
FFF	1111111111	1111111111	XXX	XXX

RE
VOA

.....

[illegible]

```

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SSSSSS
LL          II         SSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LLLLLLLLLLL IIIIIIII   SSSSSSSS
LLLLLLLLLLL IIIIIIII   SSSSSSSS

```



```
1 0001 0 MODULE RDHEDR (
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-001'
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 2
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This routine reads the desired file header, checks it for
38 0038 1 validity and correctness, and returns its address.
39 0039 1
40 0040 1 ENVIRONMENT:
41 0041 1
42 0042 1 STARLET operating system, including privileged system services
43 0043 1 and internal exec routines.
44 0044 1
45 0045 1 --
46 0046 1
47 0047 1
48 0048 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 13-Dec-1976 22:00
49 0049 1
50 0050 1 MODIFIED BY:
51 0051 1
52 0052 1 V04-001 CDS0008 Christian D. Saether 15-Sep-1984
53 0053 1 Save and restore FILE_HEADER when mapping the index
54 0054 1 file.
55 0055 1
56 0056 1 V03-008 CDS0007 Christian D. Saether 30-Aug-1984
57 0057 1 Don't update the FILE_HEADER cell if STS_LEAVE_FILEHDR
```



```
58 0058 1 is set.
59 0059 1
60 0060 1 V03-007 CDS0006 Christian D. Saether 4-Aug-1984
61 0061 1 Gee, we really ought to check the file number and
62 0062 1 sequence number also if we don't do the full
63 0063 1 check_header call.
64 0064 1
65 0065 1 V03-006 CDS0005 Christian D. Saether 9-July-1984
66 0066 1 Perform header checks if filenum = 0 (deleted header).
67 0067 1
68 0068 1 V03-005 CDS0004 Christian D. Saether 2-July-1984
69 0069 1 Only perform header checks if the buffer was read from
70 0070 1 the disk, mainly to avoid checksums on buffers already
71 0071 1 in the cache which got checksummed on the way in.
72 0072 1
73 0073 1 V03-004 ACG0408 Andrew C. Goldstein, 21-Mar-1984 11:59
74 0074 1 Fix RVN on back link when used for lock basis;
75 0075 1 interface change to CHECK_HEADER2
76 0076 1
77 0077 1 V03-003 CDS0003 Christian D. Saether 5-Mar-1984
78 0078 1 Validate lockbasis for headers.
79 0079 1
80 0080 1 V03-002 CDS0002 Christian D. Saether 30-Dec-1983
81 0081 1 Use L_NORM linkage and BIND_COMMON macro.
82 0082 1
83 0083 1 V03-001 CDS0001 Christian D. Saether 20-Sep-1983
84 0084 1 Use new MAP_IDX routine instead of calling MAP_VBN
85 0085 1 directly to avoid duplication of header serialization
86 0086 1 logic.
87 0087 1
88 0088 1 V02-004 ACG0156 Andrew C. Goldstein, 12-Mar-1980 15:23
89 0089 1 Fix header invalidation bug
90 0090 1
91 0091 1 B0103 ACG0120 Andrew C. Goldstein, 16-Jan-1980 21:18
92 0092 1 Reorder header consistency checking
93 0093 1
94 0094 1 B0102 ACG0083 Andrew C. Goldstein, 15-Nov-1979 0:59
95 0095 1 Invalidate file header if bad
96 0096 1
97 0097 1 B0101 ACG0003 Andrew C. Goldstein, 10-Nov-1978 18:55
98 0098 1 Add multi-volume support
99 0099 1
100 0100 1 B0100 ACG00001 Andrew C. Goldstein, 10-Oct-1978 20:00
101 0101 1 Previous revision history moved to [F11B.SRC]F11B.REV
102 0102 1 **
103 0103 1
104 0104 1
105 0105 1 LIBRARY 'SYSS$LIBRARY:LIB.L32';
106 0106 1 REQUIRE 'SRC$:FCPDEF.B32';
```



```
108 1097 1 GLOBAL ROUTINE READ_HEADER (FILE_ID, FCB, REALBASIS_A) : L_NORM =
109 1098 1
110 1099 1 ++
111 1100 1
112 1101 1 FUNCTIONAL DESCRIPTION:
113 1102 1
114 1103 1 This routine reads the desired file header, checks it for
115 1104 1 validity and correctness, and returns its address.
116 1105 1
117 1106 1 CALLING SEQUENCE:
118 1107 1 READ_HEADER (ARG1, ARG2, ARG3)
119 1108 1
120 1109 1 INPUT PARAMETERS:
121 1110 1 ARG1: address of file ID or 0
122 1111 1 ARG2: FCB address or 0 if none
123 1112 1 ARG3: (optional) address of cell to store real lockbasis on mismatch
124 1113 1
125 1114 1 IMPLICIT INPUTS:
126 1115 1 CURRENT_VCB contains address of VCB in process
127 1116 1
128 1117 1 OUTPUT PARAMETERS:
129 1118 1 NONE
130 1119 1
131 1120 1 IMPLICIT OUTPUTS:
132 1121 1 HEADER_LBN contains LBN of header read
133 1122 1 FILE_HEADER contains address of header buffer
134 1123 1
135 1124 1 ROUTINE VALUE:
136 1125 1 address of file header
137 1126 1
138 1127 1 SIDE EFFECTS:
139 1128 1 index file window may be turned
140 1129 1
141 1130 1 --
142 1131 1
143 1132 2 BEGIN
144 1133 2
145 1134 2 MAP
146 1135 2 FILE_ID : REF BBLOCK, ! file ID arg
147 1136 2 FCB : REF BBLOCK; ! FCB arg
148 1137 2
149 1138 2 LOCAL
150 1139 2 NOUPDFILHDR : INITIAL (0), ! flag to not update FILE_HEADER
151 1140 2 LCKFIDNUM : WORD,
152 1141 2 LCKFIDNMX : BYTE,
153 1142 2 STATUS, ! general status value
154 1143 2 VBN, ! VBN of header
155 1144 2 LBN, ! LBN of header
156 1145 2 HEADER : REF BBLOCK, ! address of header block
157 1146 2 FID : REF BBLOCK; ! local file ID pointer
158 1147 2
159 1148 2 BIND_COMMON;
160 1149 2
161 1150 2 EXTERNAL ROUTINE
162 1151 2 WRONG_LOCKBASIS : L_NORM NOVALUE,
163 1152 2 SWITCH_VOLUME : L_NORM, ! switch to correct volume
164 1153 2 MAP_IDX : L_NORM, ! map virtual to logical for index file
```



```
165      1154 2      READ_BLOCK      : L_NORM,      ! read a disk block
166      1155 2      CHECK_HEADER2    : L_NORM,      ! check header for correctness
167      1156 2      INVALIDATE      : L_NORM;      ! invalidate block buffer
168      1157 2
169      1158 2      IF TESTBITSC (STSFLGS [STS_LEAVE_FILEHDR])
170      1159 2      THEN
171      1160 2          NOUPDF;LHDR = 1;
172      1161 2
173      1162 2      ! Switch context to the volume of the specified RVN.
174      1163 2      !
175      1164 2
176      1165 2      SWITCH_VOLUME (IF .FCB NEQ 0
177      1166 2          THEN .FCB[FCB$W_FID_RVN]
178      1167 2          ELSE .FILE_ID[FID$W_RVN]
179      1168 2          );
180      1169 2
181      1170 2      ! Get the LBN of the file header. If an FCB is supplied, it contains
182      1171 2      ! the LBN. If not, derive it from the file number.
183      1172 2      !
184      1173 2
185      1174 2      LBN =
186      1175 2      BEGIN
187      1176 2          IF .FCB NEQ 0
188      1177 2          THEN .FCB[FCB$L_HDLBN]
189      1178 2          ELSE
190      1179 2              BEGIN
191      1180 2                  LOCAL
192      1181 2                      IDX_LBN,
193      1182 2                      TEMP;
194      1183 2
195      1184 2                      VBN = .FILE_ID[FID$W_NUM];
196      1185 2                      IF .CURRENT_VCB[VCB$V_EXTFID]
197      1186 2                      THEN VBN<16,8> = .FILE_ID[FID$B_NMX];
198      1187 2                      IF .VBN EQL 0 THEN ERR_EXIT (SS$NOSUCHFILE);
199      1188 2                      VBN = .VBN + .CURRENT_VCB[VCB$B_IBMAPSIZE] + .CURRENT_VCB[VCB$W_CLUSTER]*4;
200      1189 2
201      1190 2      ! Save and restore FILE_HEADER when mapping the index file. This is
202      1191 2      ! because FILE_HEADER is used in ERR_CLEANUP to determine whether to
203      1192 2      ! attempt re-reading the file header using the fib and primary_fcb.
204      1193 2      !
205      1194 2
206      1195 2          TEMP = .FILE_HEADER;
207      1196 2          IDX_LBN = MAP_IDX (.VBN);
208      1197 2          FILE_HEADER = .TEMP;
209      1198 2          .IDX_LBN
210      1199 2      END
211      1200 2      END;
212      1201 2
213      1202 2      IF .LBN EQL -1 THEN ERR_EXIT (SS$NOSUCHFILE);
214      1203 2
215      1204 2      ! Now read the header and check it for correctness. If a file ID
216      1205 2      ! was supplied, use the file number and file sequence number from
217      1206 2      ! it; else use the arguments. If the file operation is being done on a
218      1207 2      ! spooled device, the file must be marked as spooled.
219      1208 2      !
220      1209 2
221      1210 2      HEADER = READ_BLOCK (.LBN, 1, HEADER_TYPE);
```



```
222 1211 2
223 1212 2 IF .FILE_ID NEQ 0
224 1213 2 THEN
225 1214 2 FID = .FILE_ID
226 1215 2 ELSE
227 1216 2 FID = FCB[FCB$W_FID];
228 1217 2
229 1218 2 IF .STSFLGS [STS_DISKREAD]
230 1219 2 THEN
231 1220 3 BEGIN
232 1221 4 IF NOT (STATUS = CHECK_HEADER2 (.HEADER, .FID, USER_STATUS))
233 1222 3 THEN
234 1223 4 BEGIN
235 1224 4 IF .STATUS LSSU 4
236 1225 4 THEN INVALDATE (.HEADER);
237 1226 4 ERR_EXIT ();
238 1227 3 END;
239 1228 3 END
240 1229 2 ELSE
241 1230 3 BEGIN
242 1231 4 IF .HEADER[FH2$W_FID_NUM] EQL 0
243 1232 4 AND .HEADER[FH2$B_FID_NMX] EQL 0
244 1233 4 OR (.HEADER[FH2$W_FID_SEQ] NEQ .FID [FID$W_SEQ])
245 1234 3 THEN
246 1235 3 ERR_EXIT (SS$_NOSUCHFILE);
247 1236 3
248 1237 3 IF .HEADER[FH2$W_FID_NUM] NEQ .FID [FID$W_NUM]
249 1238 3 OR .HEADER[FH2$B_FID_NMX] NEQ .FID [FID$B_NMX]
250 1239 3 THEN
251 1240 4 BEGIN
252 1241 4 INVALDATE (.HEADER);
253 1242 4 ERR_EXIT (SS$_FILENUMCHK);
254 1243 3 END;
255 1244 3
256 1245 2 END;
257 1246 2
258 1247 2 IF .CLEANUP_FLAGS[CLF_SPOOLFILE]
259 1248 2 AND NOT .HEADER[FH2$V_SPOOL]
260 1249 3 AND (.HEADER[FH2$W_FID_NUM] GTRU .CURRENT_VCB[VCB$B_RESFILES]
261 1250 4 OR (.CURRENT_VCB[VCB$V_EXTFID]
262 1251 4 AND .HEADER[FH2$B_FID_NMX] NEQ 0)
263 1252 3 )
264 1253 2 THEN ERR_EXIT (SS$_NOSUCHFILE);
265 1254 2
266 1255 2 ! Validate that we have the correct lock basis for this header.
267 1256 2 !
268 1257 2
269 1258 2 LCKFIDNUM = .(LB_BASIS [.CURR_LCKINDX])<0,16>;
270 1259 2 LCKFIDNMX = .(LB_BASIS [.CURR_LCKINDX])<16,8>;
271 1260 2
272 1261 2 IF .HEADER [FH2$W_SEG_NUM] EQL 0
273 1262 2 THEN
274 1263 3 BEGIN
275 1264 4 IF .LCKFIDNUM NEQ .HEADER [FH2$W_FID_NUM]
276 1265 4 OR .LCKFIDNMX NEQ .HEADER [FH2$B_FID_NMX]
277 1266 3 THEN
278 1267 4 BEGIN
```

```

279 1268 4      IF .FCB NEQ 0
280 1269 4      THEN
281 1270 4          BUG_CHECK (XQPERR, 'wrong lock basis with fcb present');
282 1271 4
283 1272 4      ! We came in for what should be a primary header, and yet have the wrong
284 1273 4      ! lock basis. Something must be sour with the file structure so exit.
285 1274 4      ! This type of situation has probably already been caught by earlier checks
286 1275 4      ! in this routine, but we should cover ourselves here in case something
287 1276 4      ! slips by.
288 1277 4      !
289 1278 4
290 1279 4          WRONG_LOCKBASIS (.HEADER);
291 1280 4          ERR_EXIT (SS$_NOSUCHFILE);
292 1281 4          END
293 1282 3      END
294 1283 2  ELSE
295 1284 2
296 1285 2      ! This is an extension header.
297 1286 2      ! In general, we are only going to allow access to extension headers
298 1287 2      ! as a result of finding it by following links from the primary header.
299 1288 2      ! If the lock basis is wrong, we are trying to go at it directly.
300 1289 2      ! However, we will handle the case of accessing an extension header
301 1290 2      ! when the file is already accessed (which means an fcb will have been
302 1291 2      ! found). In that case, read_header is called with the optional
303 1292 2      ! realbasis_a argument which means the caller is prepared to release
304 1293 2      ! the incorrect serialization lock already acquired, acquire the correct
305 1294 2      ! lock, and read the header again.
306 1295 2      !
307 1296 2
308 1297 3      BEGIN
309 1298 3      IF .LCKFIDNUM NEQ .HEADER [FH2$W_BK_FIDNUM]
310 1299 3      OR .LCKFIDNMX NEQ .HEADER [FH2$B_BK_FIDNMX]
311 1300 3      THEN
312 1301 4          BEGIN
313 1302 4          WRONG_LOCKBASIS (.HEADER);
314 1303 4
315 1304 4          IF .FCB EQL 0 OR ACTUALCOUNT LSSU 3
316 1305 4          THEN
317 1306 4              ERR_EXIT (SS$_NOSUCHFILE);
318 1307 4
319 1308 4      ! There's a bug in the file system which screws up the RVN.
320 1309 4      ! We should get this from the FCB instead.
321 1310 4      ! Fix this when we start storing the primary FCB in all extension FCBs.
322 1311 4      !
323 1312 4
324 1313 4          (.REALBASIS_A)<0,16> = .HEADER [FH2$W_BK_FIDNUM];
325 1314 4          (.REALBASIS_A)<16,8> = .HEADER [FH2$B_BK_FIDNMX];
326 1315 4          (.REALBASIS_A)<24,8> = .HEADER [FH2$B_BK_FIDRVN];
327 1316 4          APPLY_RVN (T.REALBASIS_A)<24,8>, .CURRENT_RVN);
328 1317 4          END
329 1318 2      END;
330 1319 2
331 1320 2  HEADER_LBN = .LBN;          ! return LBN of header
332 1321 2  IF NOT .NOUPDFILHDR
333 1322 2  THEN
334 1323 2      FILE_HEADER = .HEADER;    ! and address
335 1324 2  RETURN .HEADER;            ! and the header itself
```


! end of routine READ_HEADER

				.TITLE	RDHEDR	
				.IDENT	\V04-001\	
				.EXTRN	WRONG_LOCKBASIS	
				.EXTRN	SWITCH_VOLUME, MAP_IDX	
				.EXTRN	READ_BLOCK, CHECK_HEADER2	
				.EXTRN	INVAIDATE, BUG\$_XOPERR	
				.PSECT	\$CODE\$,NOWRT,2	
			003C 00000	.ENTRY	READ_HEADER, Save R2,R3,R4,R5	: 1097
			55 D4 00002	CLRL	NOUPDFILHDR	: 1132
03	A6	AA	03 E5 00004	BBCC	#3, -90(BASE), 1\$	: 1158
		55	01 D0 00009	MOVL	#1, NOUPDFILHDR	: 1160
		50	08 AC D0 0000C 1\$:	MOVL	FCB, R0	: 1165
			06 13 00010	BEQL	2\$	
		7E	28 A0 3C 00012	MOVZWL	40(R0), -(SP)	: 1166
			08 11 00016	BRB	3\$	
		50	04 AC D0 00018 2\$:	MOVL	FILE_ID, R0	: 1167
		7E	04 A0 3C 0001C	MOVZWL	4(R0), -(SP)	
	0000G	CF	01 FB 00020 3\$:	CALLS	#1, SWITCH_VOLUME	: 1165
		50	08 AC D0 00025	MOVL	FCB, R0	: 1176
			06 13 00029	BEQL	4\$	
		54	34 A0 D0 0002B	MOVL	52(R0), LBN	: 1177
			3F 11 0002F	BRB	6\$	
		52	04 AC D0 00031 4\$:	MOVL	FILE_ID, R2	: 1184
		51	62 3C 00035	MOVZWL	(R2), VBN	
		50	98 AA D0 00038	MOVL	-104(BASE), R0	: 1185
51	06	0B	05 E1 0003C	BBC	#5, 11(R0), 5\$	
	08	A0	05 A2 F0 00041	INSV	5(R2), #16, #8, VBN	: 1186
		10	51 D5 00047 5\$:	TSTL	VBN	: 1187
			73 13 00049	BEQL	11\$	
		50	98 AA D0 0004B	MOVL	-104(BASE), R0	: 1188
		52	38 A0 9A 0004F	MOVZBL	56(R0), R2	
		52	51 C0 00053	ADDL2	VBN, R2	
		50	3C A0 3C 00056	MOVZWL	60(R0), R0	
		51	6240 DE 0005A	MOVAL	(R2)[R0], VBN	
		52	04 AA D0 0005E	MOVL	4(BASE), TEMP	: 1195
			51 DD 00062	PUSHL	VBN	: 1196
	0000G	CF	01 FB 00064	CALLS	#1, MAP_IDX	
	04	AA	52 D0 00069	MOVL	TEMP, 4(BASE)	: 1197
		54	50 D0 0006D	MOVL	IDXLBN, LBN	: 1198
	FFFFFFFF	8F	54 D1 00070 6\$:	CMPL	LBN, #-1	: 1202
			45 13 00077	BEQL	11\$	
		7E	01 7D 00079	MOVQ	#1, -(SP)	: 1210
			54 DD 0007C	PUSHL	LBN	
	0000G	CF	03 FB 0007E	CALLS	#3, READ_BLOCK	
		52	50 D0 00083	MOVL	R0, HEADER	
			04 AC D5 00086	TSTL	FILE_ID	: 1212
			06 13 00089	BEQL	7\$	
		53	04 AC D0 0008B	MOVL	FILE_ID, FID	: 1214
			05 11 0008F	BRB	8\$	
53	08	AC	24 C1 00091 7\$:	ADDL3	#36, FCB, FID	: 1216
		1C	A6 AA E9 00096 8\$:	BLBC	-90(BASE), 10\$	: 1218

		80	AA	9F	0009A	PUSHAB	-128(BASE)	1221	
			0C	BB	0009D	PUSHR	#^M<R2,R3>		
0000G	CF		03	FB	0009F	CALLS	#3, CHECK_HEADER2		
	39		50	E8	000A4	BLBS	STATUS, 14\$		
	04		50	D1	000A7	CMPL	STATUS, #4	1224	
			07	1E	000AA	BGEQU	9\$		
			52	DD	000AC	PUSHL	HEADER	1225	
0000G	CF		01	FB	000AE	CALLS	#1, INVALIDATE		
			00	BF	000B3	CHMU	#0	1226	
				04	000B5	RET			
		08	A2	B5	000B6	TSTW	8(HEADER)	1231	
			05	12	000B9	BNEQ	12\$		
		0D	A2	95	000BB	TSTB	13(HEADER)	1232	
			76	13	000BE	BEQL	18\$		
02	A3	0A	A2	B1	000C0	CMPW	10(HEADER), 2(FID)	1233	
			6F	12	000C5	BNEQ	18\$		
	63	08	A2	B1	000C7	CMPW	8(HEADER), (FID)	1237	
			07	12	000CB	BNEQ	13\$		
05	A3	0D	A2	91	000CD	CMPB	13(HEADER), 5(FID)	1238	
			0C	13	000D2	BEQL	14\$		
			52	DD	000D4	PUSHL	HEADER	1241	
0000G	CF		01	FB	000D6	CALLS	#1, INVALIDATE		
		08B0	8F	BF	000DB	CHMU	#2224	1242	
				04	000DF	RET			
			6A	95	000E0	TSTB	(BASE)	1247	
			1D	18	000E2	BGEQ	15\$		
18	35	A2	04	E0	000E4	BBS	#4, 53(HEADER), 15\$	1248	
		50	98	AA	D0	MOVL	-104(BASE), R0	1249	
		51	4F	A0	9A	MOVZBL	79(R0), R1		
	08	A2	51	B1	000F1	CMPW	R1, 8(HEADER)		
			5E	1F	000F5	BLSSU	21\$		
05	0B	A0	05	E1	000F7	BBC	#5, 11(R0), 15\$	1250	
			0D	A2	95	000FC	TSTB	13(HEADER)	1251
			54	12	000FF	BNEQ	21\$		
	50		14	AA	D0	00101	MOVL	20(BASE), R0	1258
	53	0080	CA40	F7	00105	CVTLW	128(BASE)[R0], LCKFIDNUM		
	50		14	AA	D0	0010B	MOVL	20(BASE), R0	1259
	51	0082	CA40	F6	0010F	CVTLB	130(BASE)[R0], LCKFIDNMX		
		04	A2	B5	00115	TSTW	4(HEADER)	1261	
			1E	12	00118	BNEQ	19\$		
08	A2		53	B1	0011A	CMPW	LCKFIDNUM, 8(HEADER)	1264	
			06	12	0011E	BNEQ	16\$		
0D	A2		51	91	00120	CMPB	LCKFIDNMX, 13(HEADER)	1265	
			67	13	00124	BEQL	24\$		
		08	AC	D5	00126	TSTL	FCB	1268	
			04	13	00129	BEQL	17\$		
				FEFF	0012B	BUGW		1270	
				0000*	0012D	.WORD	<BUG\$ XQPERR!4>		
			52	DD	0012F	PUSHL	HEADER	1279	
0000G	CF		01	FB	00131	CALLS	#1, WRONG_LOCKBASIS		
			1D	11	00136	BRB	21\$	1280	
42	A2		53	B1	00138	CMPW	LCKFIDNUM, 66(HEADER)	1298	
			06	12	0013C	BNEQ	20\$		
47	A2		51	91	0013E	CMPB	LCKFIDNMX, 71(HEADER)	1299	
			49	13	00142	BEQL	24\$		
			52	DD	00144	PUSHL	HEADER	1302	
0000G	CF		01	FB	00146	CALLS	#1, WRONG_LOCKBASIS		

			08	AC	D5	0014B	TSTL	FCB		1304
				05	13	0014E	BEQL	21\$		
		03		6C	91	00150	CMPB	(AP), #3		
				05	1E	00153	BGEQU	22\$		
			0910	8F	BF	00155	CHMU	#2320		1306
					04	00159	RET			
					B0	0015A	MOVW	66(HEADER), @REALBASIS_A		1313
					F0	0015F	INSV	71(HEADER), #16, #8, @REALBASIS_A		1314
					F0	00166	INSV	70(HEADER), #24, #8, @REALBASIS_A		1315
					D0	0016D	MOVL	REALBASIS_A, R0		1316
					95	00171	TSTB	3(R0)		
					12	00174	BNEQ	23\$		
					90	00176	MOVB	-96(BASE), 3(R0)		
					D0	0017B	MOVL	REALBASIS_A, R0		
					91	0017F	CMPB	3(R0), #1		
					12	00183	BNEQ	24\$		
					D5	00185	TSTL	-96(BASE)		
					12	00188	BNEQ	24\$		
					94	0018A	CLRB	3(R0)		
					D0	0018D	MOVL	LBN, -80(BASE)		1320
					E8	00191	BLBS	NOUPDFILHDR, 25\$		1321
					D0	00194	MOVL	HEADER, 4(BASE)		1323
					D0	00198	MOVL	HEADER, R0		1324
					04	0019B	RET			1325

; Routine Size: 412 bytes, Routine Base: \$CODE\$ + 0000

:	337	1326	1
:	338	1327	1 END
:	339	1328	0 ELUDOM

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	412	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	38 0	1000	00:01.9

RDHEDR
V04-001

K 9
16-Sep-1984 01:01:13
14-Sep-1984 16:09:09

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11X.SRC]RDHEDR.B32;2 Page 10
(2)

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:RDHEDR/OBJ=OBJ\$:RDHEDR MSRC\$:RDHEDR/UPDATE=(ENHS:RDHEDR)

: Size: 412 code + 0 data bytes
: Run Time: 00:22.5
: Elapsed Time: 00:52.8
: Lines/CPU Min: 3538
: Lexemes/CPU-Min: 42141
: Memory Used: 300 pages
: Compilation Complete

0172 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RWB
LIS

RDBLOK
LIS

REQUL
LIS

RWATR
LIS

REMOVE
LIS

RDHEDR
LIS

RETDIR
LIS